Predict

Re-usable, Scalable Prediction Methodology  
*Methods described are model agnostic and output agnostic – Can be used over any model type, and any output type.*

*Output methods (PDP/SHAP) are intended to give decision makers information about ‘where’ an intervention might be most impactful.*

*All methods described (from explore -> output) have supported libraries in both R / Python.*

# Exploration / Preprocessing

**Quick Notes:**

* Data Exploration
  + ‘Describe’
* Preprocessing
  + Ex: Impute Missing Values, Data Types Conversion, Splitting Strings
* Basic Feature Engineering
  + Row-level calculations (ex: Interactions, encoding, counts)
  + Avoid creating aggregate variables here – these variables should be created *after* train/test split, to avoid test set data leakage

**Tip:** Inspect value counts of variables, distinct counts of observation ID *often*

# Split Data / Train Model / Cross Validation

**Split Data:**

* Split data into Train / Test sets
  + Test set should represent ~20-33% of entire dataset
  + Test set should remain unused / untouched during model selection / tuning stage

**Create Cross-Observation Features:**

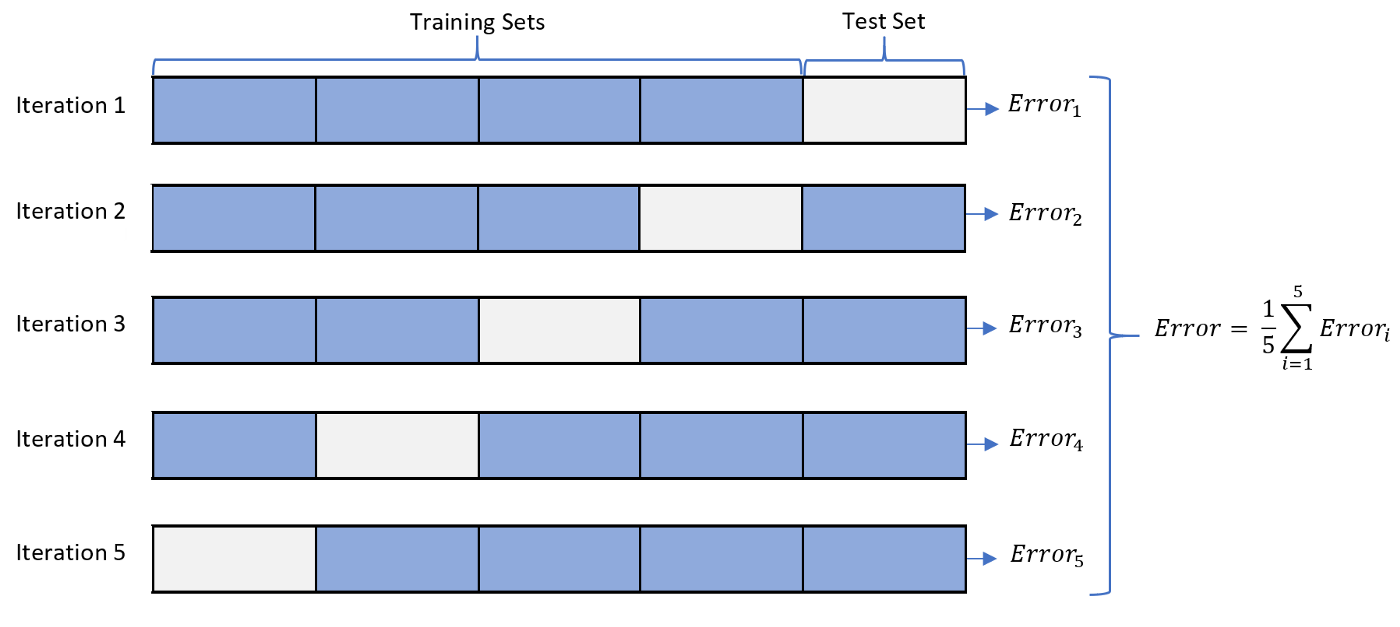
* Common to create features that use information from other examples (such as a School District Median of assessment scores).
* However, creating these metrics over then entire set (including test) can cause the test set to implicitly inform the training set, leading to a biased model.

**Train Model:**

* Choose model appropriate for desired output – Classifier if binary, Regression if continuous
  + Tree-based models (boosted trees or Random Forest) tend to work well for most datasets, and include several parameters to prevent overfitting

**Cross Validation:**

* Best practice is Cross Validation to get scores on your trained model

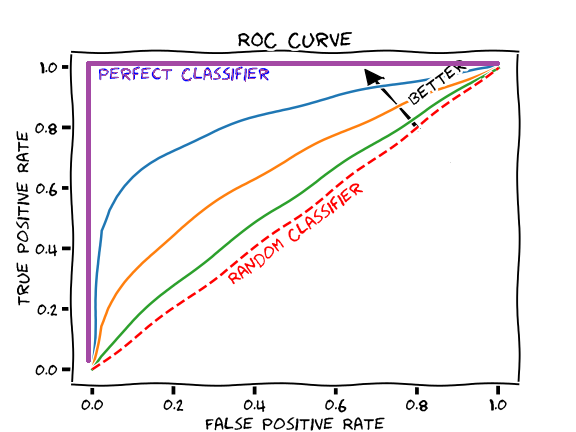
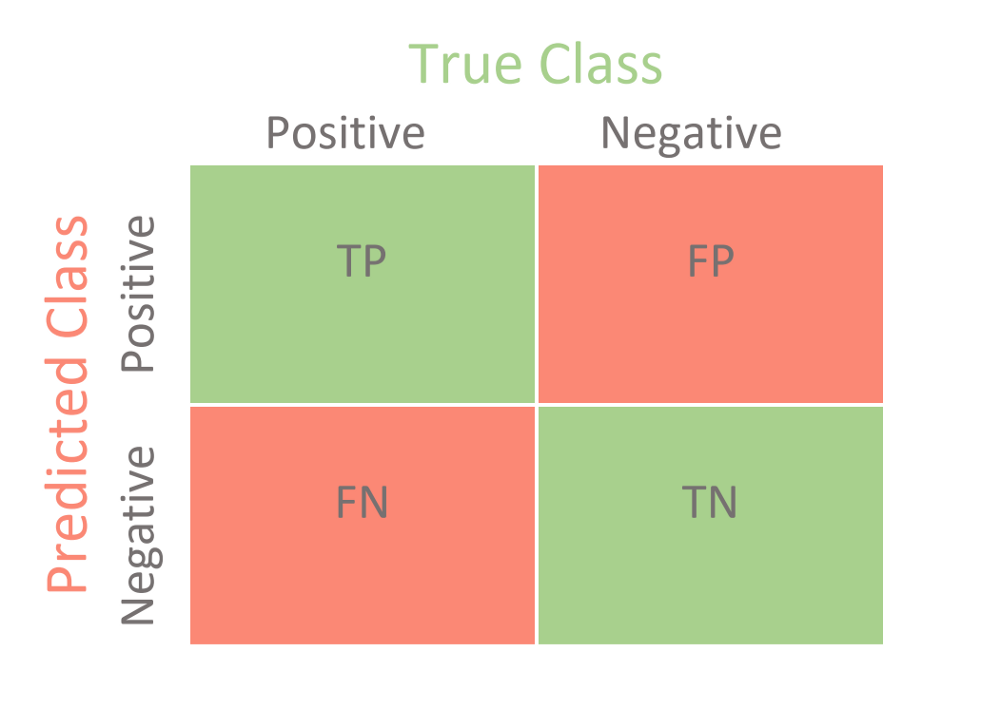


* + Cross validation ensures that you get approximate generalizable model performance without generating scores based on the test set.
  + Average error across K-Folds gives the estimate of generalizable error

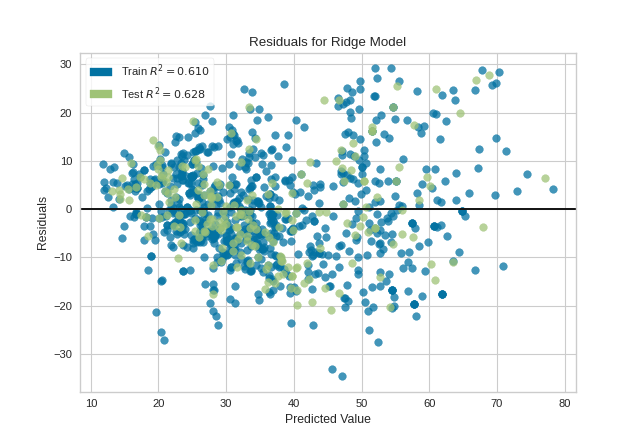
# Model Selection / Tuning

**Model Selection:**

* Inspect model performance (performance measured by average Cross Validation score of chosen performance metric).
  + Does it perform significantly better than some baseline model? (i.e., predicting majority class in binary classification).
  + Is the prediction distribution similar to the true distribution of the outcome? If not, model is likely biased.
* Classification Techniques
  + Confusion Matrix, ROC Curve (True Positive Rate vs False Positive Rate)
  + These can answer questions: where is my model struggling? Where is it performing well? At what prediction threshold do I get the best results?



* Regression Techniques
  + Residual Plot, Prediction Error Plot (Y vs Y-hat plot)
  + Answers questions: are my residuals normally distributed, or is there a pattern? A strong non-random pattern would indicate an underfitted model (leaving important variables out). A prediction error plot can at a glance show how well our model fits the truth, and where the predictions differ the most.



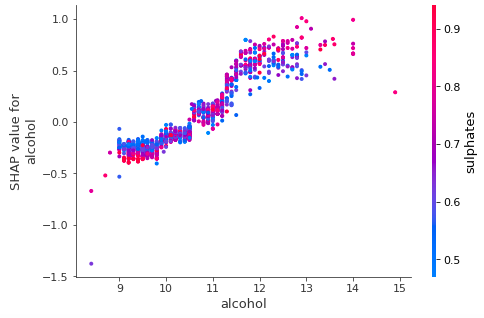
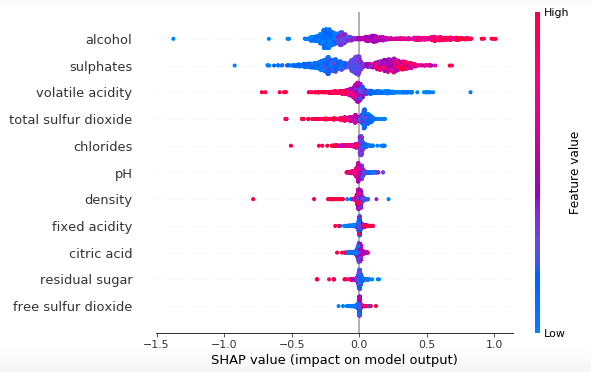
**Tuning:**

* Once you have a reasonable model, hyperparameter tuning (or tweaking model settings, such as number of leaves per branch in random forest) can improve model performance.
  + \*Note that typically the biggest model performance gains will come from data improvements such as feature engineering, or simply getting more/more useful data.
  + Tuning should always be done within the training set, using cross validation – never tweak your parameters based on test set results, as this could lead to overfitting & less generalizable predictions.

# Model Output

**SHAP values:**

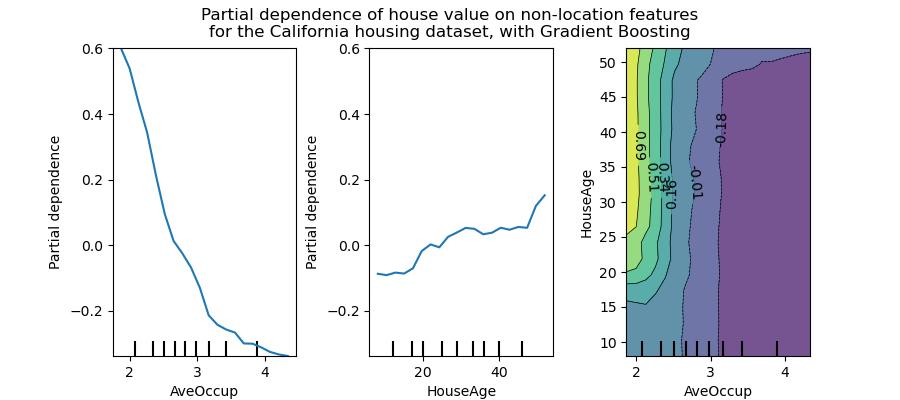
* SHAP or Shapley Additive Values, are a method of quantifying feature impact at the individual level (row by row).
* These values have a nice property:
  + SUM(SHAP values of features) = prediction – prediction of baseline
* There are several interesting plots to answer:
  + Which features are most important to prediction (for each observation, and in aggregate)? We can see this in the summary plot – below, left (top features descending).
  + At which feature values (hi/low) does the affect on the model differ?
  + Is the SHAP value for x mediated by y? (with dependence plot – below, right)



*Example target is wine quality rating, credit* [*https://towardsdatascience.com/explain-your-model-with-the-shap-values-bc36aac4de3d*](https://towardsdatascience.com/explain-your-model-with-the-shap-values-bc36aac4de3d)

**Partial Dependence Plots:**

* PDPs represent how much a feature affects the outcome, but also at **which values of the feature** – represents marginal effects.
* In this way it can capture non-linear effects of features and can be used to investigate possible “process” variables that decision makers could alter, and ‘where’ the biggest impact of intervention would be achieved.
* An example, showing PDP for 2 features, Occupancy/Age, and the interaction between the two. Each plot can potentially identify decision boundaries that have real value (as illustrated, occupancy reduces price, but has less of an effect at age > 15).



* How it works:

Diagram

Description automatically generated

**Feature Importance & Effect Sizes:**

* Log Odds & Regression Effects:
  + These are useful in contexts where decision boundary doesn’t matter as much – For example, when you have data that fits a linear model, and the suitable output is P(outcome | feature1) is 1.5x more likely.
* Another potential method of calculating feature importance is trees. However, these give a numerical output – either ‘# of times variable was split’ / ‘uncertainty reduced by splitting on variable’
  + Since these answer ‘Which features were most important’? without suggestion direction or at which values, this approach is probably more suited to building the model

# Resources / References

**Resources/References:**

* <https://scikit-learn.org/stable/modules/partial_dependence.html>
* <https://www.kaggle.com/dansbecker/advanced-uses-of-shap-values>
* <https://towardsdatascience.com/>
* <https://bgreenwell.github.io/pdp/articles/pdp.html>